COMP2121 Tutorial 6: Interrupts

1. The EICRA register is used to indicate what condition should be present for external interrupts to occur, and looks like this:

Bit	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

where each pair of bits ISCn1 and ISCn0 mean the following for INTn:

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INTn generates asynchronously an interrupt request.
1	1	The rising edge of INTn generates asynchronously an interrupt request.

The EIMSK register is used to enable the external interrupts and looks like this:

Bit	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

In "m64def.inc", the values in these registers have been defined to their bit value. e.g., ISC00 = 0, ISC11=3 and INT2=2. Knowing this, examine the following code:

```
.def temp=r16
ldi temp, (0b10 << ISC00) | (0 << ISC10) |(0b11 << ISC20)
sts EICRA, temp
ldi temp, (1 << INT0) | (1 << INT2)
out EIMSK, temp
sei
```

a) What is the value (in binary) that is written to the EICRA register?
b) Why do we use this approach to set up the register values?
c) Which external interrupts can occur, and when will they occur?
d) What is the difference between the 'sts' instruction and the 'out' instruction?

2. This question looks at the registers associated with PORT A. The following tables might help:

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port A Data Direction Register – DDRA

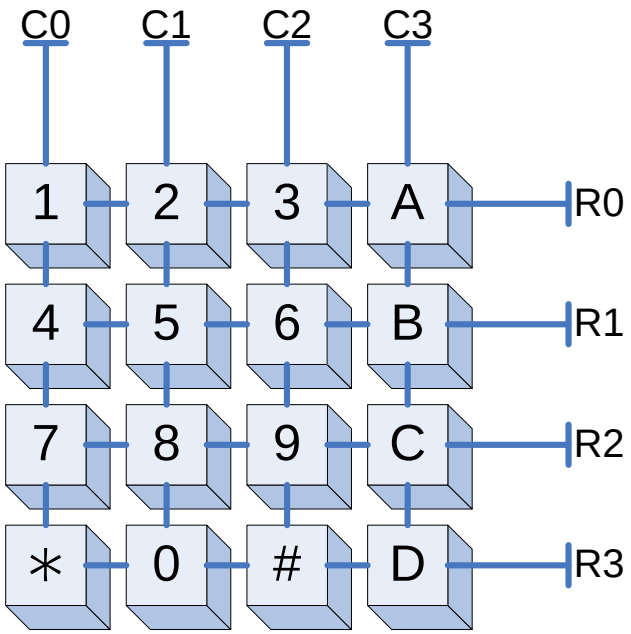
Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

a) What is the purpose of the DDRA register?
b) What is the purpose of PORTA0 when DDA0 = 1?
c) What is the purpose of PORTA0 when DDA0 = 0 and PUD = 0?
d) What is the purpose of the PINA register?

3. The Keypad on the AVR boards is a set of 16 push buttons. The keypad has four rows and four columns, accessible via the pins R0-R3 and C0-C3. When you push a button on the keypad, it connects the column of the key to the row of the key as follows:



One method to correctly read what keys are being pressed is to:

1: Set up the rows so that they read a Logic 1 when none of the buttons on the row is pushed.
2: Set one column to Logic 0 and all other columns to Logic 1.
3: Read the values of the row pins. If a row reads as Logic 0, you know that the switch at that row and column must be pushed.
4: Set a different column to Logic 0 and read the rows.
5: Repeat steps 3 and 4 until a switch is found to be pressed or you run out of columns.
6: Repeat steps 2-5 again if you want to see whether a different switch is pushed.

Part of your third lab requires you to perform this algorithm. Steps 2-5 should be fairly simple to code, but step 1 is not so obvious. The way to accomplish this is with pull-up resistors. A pull-up resistor ties an input pin to Logic 1 via a resistor. This means that an input pin will still read any value that is input, and will read Logic 1 if disconnected.

To further understand this, look at switch 5 in the above diagram. When none of the switches connected to row 1 are pushed, the circuit (with pull-up shown) looks like this:

If read, the port would read a Logic 1 via the pull-up resistor.

When switch 5 is pushed, the circuit looks like this:

In this case, the port connected to R1 will always read the current value of C1. When C1 is Logic 0 there will be a voltage drop across the resistor, but this will not affect the value being read. Thus, the pull-up resistor accomplishes the desired task.

a) How do you setup an AVR I/O port so that it has pull-up resistors connected to its input pins? (See question 2 of this tute)

b) Write the code to find a switch that has been pushed by scanning either the columns or rows. (You have to do this for your lab, anyway)

c) Can you see an electrical problem with this scanning method when two switches on the same row are pushed at the same time (e.g., 5 and 6)? How could you correct this? (Hint: There might be something better you can do than output logic 1s to the columns you are not testing.)